may be available to the "value portfolio" worker **155** as well as all other jobs **182-1** to **182**-N running on all other node computers **800-1** to **800**-N within a particular service. Then, according to the "value portfolio" worker **155**, one thousand separate descendant jobs **182-1** to **182-1000** named, for example, "value instrument no. **1**," "value instrument no. **2**," etc., are divided out and sent to the scheduler **600** for assignment to an available node computer **800** within the service. The one thousand descendant jobs **182-1** to **182-1000** may each be sent to and processed on available node computers **800-1** to **800**-N. During processing, each of the descendant jobs **182-1** to **182-1000** has access to the market environment results computed earlier and stored in the global cache **900**. As a result, the descendant jobs **182-1** to **182-1000** may not need to perform the yield curve computation themselves and may not need to contact the calling application **180** for such information, but rather, can more quickly obtain the results of the yield curve computation stored in the global cache **900**. Upon completion of each of the one thousand descendant jobs **182-1** to **182-1000**, the "value portfolio" job **182** aggregates the outputs from the "value instrument" jobs **182-1** to **182-1000** for further computation of a portfolio value result.

H. Method of Troubleshooting/Debugging One Embodiment of a System

One embodiment of the system **10** also has additional functionality that may allow a worker **155** to be deployed on a local computer **100** without accessing the compute backbone **300** infrastructure or the network **200**. To allow an applications developer **30** to debug its worker modules **195-1** to **195**-N locally on its local computer **100** (which, in one embodiment, is the development host for the applications developer **30**), the compute backbone **300** is capable of (i) providing a simplified replica of itself, including an API **190**, and (ii) initializing worker modules **195-1** to **195**-N in the same process space in which the calling application **180** resides. Such a capability may enable an applications developer **30** to debug functionality, such as persistence and parameter passing, in an environment where the developer **30** has access to all necessary information about both the calling application **180** and the environment on which it is running (i.e., the replicated functionality of the compute backbone **300**). For example, if a worker module **195** performs properly on the local computer **100**, it will also perform properly when deployed on the compute backbone **300**.

FIG. **11** illustrates certain operations performed in one embodiment of a method of running a calling application **180** in local mode. For any particular calling application **180**, an applications developer **30** may create both a worker module **195** and one or more jobs **182** (step **1910**). At initialization, the developer **30** links the calling application **180** to the API **190** file associated with local mode operation (as opposed to the API **190** file associated with network mode operation) (step **1920**). The API **190** then loads the worker module **195** into the process space of the local computer **100** (step **1930**). The API **190** ensures that a replica of all major functions performed by the compute backbone **300** (e.g., scheduling, caching, etc.) are loaded into the data storage devices **110-1** to **110**-N of the local computer **100** (step **1940**). The worker **155** is then processed on the CPU **120** of the local computer **100** (step **1950**). Unlike the parallel computing operation of network mode on the actual compute backbone **300** infrastructure, processing in local mode is accomplished sequentially, or perhaps concurrently if multithreading is used.

Although illustrative embodiments and example methods have been shown and described herein in detail, it should be noted and will be appreciated by those skilled in the art that there may be numerous variations and other embodiments which may be equivalent to those explicitly shown and described. For example, the scope of the present invention is not necessarily limited in all cases to execution of the aforementioned steps in the order discussed. Unless otherwise specifically stated, the terms and expressions have been used herein as terms and expressions of description, not of limitation. Accordingly, the invention is not limited by the specific illustrated and described embodiments and examples (or the terms or expressions used to describe them) but only by the scope of appended claims.

What is claimed is:

1. A system, comprising:

a processor;

a transaction manager in communication with a plurality of local computers, wherein said transaction manager supports a plurality of message protocols to enable communication between said transaction manager and said plurality of local computers, wherein each said local computer runs a calling application;

a queue in communication with said transaction manager, wherein said queue receives and stores a plurality of jobs and task inputs from said transaction manager, sends a plurality of computation requests to one or more of a plurality of node computers, and provides computation results when polled by said transaction manager;

a scheduler in communication with said queue, wherein said scheduler routes a plurality of incoming tasks to a plurality of compute functions on the plurality of node computers and allocates computing resources of said plurality of node computers,

a service manager in communication with said transaction manager, said scheduler, and said plurality of node computers, wherein said service manager controls allocation of computing resources among a plurality of users of said plurality of local computers such that a failure of one of said calling applications of one of said users will not adversely affect another of said calling applications of another of said users, even if both applications are running simultaneously; and

a cache in communication with said plurality of node computers, wherein a minimum availability of said system is defined by an availability of said queue such that addition of one or more devices to said system downstream of said queue does not increase a failure probability of said system.

2. The system of claim **1**, wherein said transaction manager facilitates delivery of each of said plurality of computation requests from said calling applications to said system.

3. The system of claim **2**, wherein said transaction manager guarantees delivery of each of said plurality of computation requests from said calling applications to said system.

4. The system of claim **1**, wherein each of said calling applications is created in a different programming language.

5. The system of claim **1**, wherein communications between said transaction manager and each of said plurality of local computers are secure and involve an authentication process before access to said system is granted.

6. The system of claim **1**, wherein all information pertinent for a particular one of said plurality of jobs is stored persistently in said queue at least until said job has been completed or has expired.

* * * * *